WHAT IS CLAIMED IS:

1. A computer-implemented method for processing rendering data, comprising:

transforming the rendering data containing vertices from model space into clip space; and

storing each of the vertices in a vertex cache as needed to facilitate a single streamline branched architecture that avoids processing duplication of the vertices.

2. The computer-implemented method of claim 1, further comprising examining each of the vertices before lighting to determine whether to cull.

3. The computer-implemented method of claim 2, further comprising discarding any vertices that are culled.

4. The computer-implemented method of claim 3, further comprising continuing processing of any vertices that are not culled.

5. The computer-implemented method of claim 1, further comprising generating coordinates for the vertices by performing lighting and texture generation and transformation.

6. The computer-implemented method of claim 5, further comprising performing view frustum clipping on the coordinates after the lighting and texture generation and transformation.

7. The computer-implemented method of claim 6, wherein the coordinates are normalized homogenous coordinate system (NHCS) clip space coordinates.

8. The computer-implemented method of claim 1, further comprising using Direct3D for mobile as a rendering standard.

9. The computer-implemented method of claim 1, wherein the vertex cache in implemented in software instead of hardware.

10. A computer-readable medium having computer-executable instructions for performing the computer-implemented method recited in claim 1.

11. A process for transforming and lighting rendering data, comprising:

inputting rendering data in model space containing vertices;

transforming the rendering data from model space to clip space;

determining whether to cull at least some of the vertices prior to lighting the rendering data; and

lighting each of the vertices to compute color.

12. The process as set forth in claim 11, wherein determining whether to cull further comprises examining each of the vertices.

13. The process as set forth in claim 12, further comprising determining whether a vertices forms a back face of a triangle.

14. The process as set forth in claim 13, further comprising discarding the vertices if it does form the back face of a triangle

15. The process as set forth in claim 13, further comprising keeping the vertices if it does not form the back face of a triangle.

16. The process as set forth in claim 12, further comprising determining whether a vertices is outside of one view frustum clip plane.

17. The process as set forth in claim 16, further comprising discarding the vertices if it is outside of one view frustum clip plane.

18. The process as set forth in claim 16, further comprising keeping the vertices if it is not outside of one view frustum clip plane.

19. One or more computer-readable media having computer-readable instructions thereon which, when executed by one or more processors, cause the one or more processors to implement the process of claim 11.

20. A computer-readable medium having computer-executable instructions for rendering graphics on an embedded device, comprising:

inputting 3D data containing vertices in model space;

transforming the 3D data into clip space;

examining each of the vertices before lighting to determine whether to cull the vertices;

storing the vertices as needed in a vertex cache to provide a single streamline branched architecture that avoids processing duplication of the vertices; and

performing view frustum clipping of the vertices to generate an output of 2D screen coordinates.

21. The computer-readable medium of claim 20, wherein examining each of the vertices before lighting further comprises determining whether any of the vertices form a back face of a triangle and, if so, culling those vertices.

22. The computer-readable medium of claim 21, further comprising discarding each of the culled vertices and continue processing vertices that have not been culled.

23. The computer-readable medium of claim 20, wherein examining each of the vertices before lighting further comprises determining whether any of the vertices are outside of one view frustum clip plane and, if so, culling those vertices.

24. The computer-readable medium of claim 23, further comprising discarding each of the culled vertices and continue processing vertices that have not been culled.

25. The computer-readable medium of claim 20, wherein the vertex cache is contained in software and not in hardware.

26. The computer-readable medium of claim 20, wherein the view frustum clipping is performed after a lighting and texture generation and transformation of the vertices.

27. The computer-readable medium of claim 20, wherein the view frustum clipping is performed on normalized homogenous coordinate system (NHCS) clip space coordinates of each of the vertices.

28. A transform and lighting module for preparing rendering data for rendering on an embedded computing device, comprising:

a transformation module that transforms the rendering data into clip space;

a vertex cache that stores vertices contained in the rendering data; and

a lighting module that computes color for each of the vertices; and

a culling module positioned before the lighting module that examines each of the vertices to determine whether to send vertices to the lighting module.

29. The transformation and lighting module as set forth in claim 28, wherein the vertex cache is implemented in software.

30. The transformation and lighting module as set forth in claim 28, wherein the culling module performs at least one of: (a) back face culling; (b) view frustum culling.

31. The transformation and lighting module as set forth in claim 28, further comprising a view frustum clipping module positioned after the lighting module and after a texture generation and transformation module.

SOFTWARE-IMPLEMENTED TRANSFORM AND LIGHTING MODULE AND PIPELINE FOR GRAPHICS RENDERING ON EMBEDDED PLATFORMS USING A FIXED-POINT NORMALIZED HOMOGENOUS COORDINATE SYSTEM

ABSTRACT OF THE DISCLOSURE

A software-implemented transform and lighting module and pipeline designed and optimized for embedded platforms (such as mobile computing devices). The transform and lighting module and pipeline includes a number of features that make it well-suited for use on embedded devices. These features include a single streamline branched architecture that allows efficient processing on a CPU of an embedded device and saves computational time. This architecture is facilitated by use of a vertex cache that stores vertices as needed to avoid duplication in processing of the vertices. A culling feature culls vertices before lighting instead of lighting all vertices. A back face culling technique examines each of the vertices to determines whether a back face of a triangle is formed. If so, then the vertex is culled. A second technique involved determining whether a vertex is outside of one view frustum clip plane. If so, then the vertex is culled.